Übungsblatt 2

OPR2UE, Objektorientierte Programmierung, UE

Erreichbare Punkte	24 Punkte
Letzter Abgabetermin	Donnerstag, 6. April 2017, 23:55 Uhr
Abgabedokumente	Quelldateien inkl. Tests
Abgabeort	Moodle (https://hagenberg.elearning.fh-ooe.at)

Aufgabe 1 - Passwortsicherheit 6 Punkte

Gute Passwörter sollten so aufgebaut sein, dass sie nicht einfach mit Brute-Force-Attacken erraten werden können.

Eure Aufgabe ist es, ein Prüfprogramm `PasswordChecker` zu verfassen, in welchem ein oder mehrere Passwörter beim Programmstart über die Kommandozeile übergeben wird. Die Passwörter sind dann auf gängige Richtlinien für sichere Passwörter zu überprüfen.

Ein Passwort sollte zumindest folgende Voraussetzungen erfüllen:

- Das Passwort sollte mindestens acht Zeichen lang sein.
- Es sollen mindestens zwei Buchstaben und mindestens zwei Ziffern enthalten sein.
- Unter den Buchstaben im Passwort sollte mindestens ein Groß- und mindestens ein Kleinbuchstabe enthalten sein.
- Darüber hinaus sollte mindestens ein Sonderzeichen (!"#$%&'()*+,-./:;<=>?@) enthalten sein.
- Mindestens eine Ziffer oder ein Sonderzeichen sollen innerhalb des Passworts vorkommen.

„zX27$afF0“ erfüllt beispielsweise diese Kriterien, „sonne123“ aber nicht (kein Sonderzeichen und kein Großbuchstabe).

Das Programm soll das übergebene Passwort ausgeben sowie alle Regeln, die durch das Passwort verletzt werden. Entspricht das Passwort den Regeln, soll ein dementsprechender Hinweis ausgegeben werden.

Hinweis: Die ASCII-Zeichentabelle ist für die Lösung dieses Beispiels sehr hilfreich.

Beispiele:

```
You entered the following password: sonne123
The password does not contain at least one minuscule and one capital.
The password does not contain at least one special character.

You entered the following password: zX27$afF0
The password complies with the rules.
```

Aufgabe 2 - Datum prüfen 10 Punkte

a) Implementiere eine Datumsvalidierung 6 Punkte

Entwickle eine Methode, die prüft, ob ein gegebenes Datum im gregorianischen Kalender repräsentiert. Die Methode übernimmt die Werte für Tag, Monat und Jahr und überprüft diese dann auf Korrektheit.

Beachte dabei folgende Regeln:

- Der Wert für den Monat muss zwischen 1 und 12 liegen.
- Es gibt Monate mit 31, 30 und 28 Tagen (in Schaltjahren auch mit 29 Tagen).
- Das Datum muss innerhalb der gregorianischen Zeitrechnung liegen, d. h. nach dem 15. 10. 1582 sein.

b) Teste die Implementierung 2 Punkte

Entwickle Unit-Tests für die Implementierung. Beachte die folgenden Punkte:

- Die Tests sollen aus den Anforderungen abgeleitet werden.
- Die Tests sollen die möglichen Gruppen von Eingabeparametern berücksichtigen.
- Fehler- und Sonderfälle sollen von den Tests abgedeckt werden.

c) Entwickle eine Konsolen-Benutzerschnittstelle 2 Punkte

Schreib ein Konsolenprogramm, welches über Benutzereingaben das Datum einliest, validiert und anschließend den Benutzer über das Prüfungsergebnis informiert.

Anmerkung: Beim Einlesen der Datumsparameter von der Kommandozeile kann davon ausgegangen werden, dass ganz Zahlen eingegeben werden.

Beispiel 1:

```
Enter day value: 14
Enter month value: 10
Enter year value: 1582
The date value 14/10/1582 is not correct!
```

Beispiel 2:

```
Enter day value: 21
Enter month value: 3
Enter year value: 2007
The date value 21/3/2007 is a correct date!
```

Beispiel 3:

```
Enter day value: 29
Enter month value: 2
Enter year value: 2000
The date value 29/2/2000 is a correct date!
```

Beispiel 4:

```
Enter day value: 45
Enter month value: 3
Enter year value: 2020
The date value 45/3/2020 is not correct!
```

Aufgabe 3 - Newton'sches Verfahren 8 Punkte

a) Implementiere die Wurzelberechnung 5 Punkte

Die Wurzel einer Zahl kann näherungsweise mit Hilfe des Newton'schen Verfahrens berechnet werden.

Dieses Approximationsverfahren geht davon aus, dass der Wert x der Quadratwurzel einer Zahl n nicht bekannt ist. Das Verfahren startet mit einem Schätzwert q > 0 für x.

Deine Aufgabe ist es, ein Programm zu schreiben, das dieses Verfahren implementiert. Diesem Programm wird eine positive ganze Zahl (ohne Null) über die Konsole (mit Hilfe der Klasse `Scanner`) übergeben. Der Wert der Quadratwurzel soll bis auf eine Abweichung von 6 Kommastellen berechnet werden ($|n - q^2| < 0{,}000\,001$). Zur Berechnung des Betrags kann die Bibliotheksfunktion `Math.abs(double n)` verwendet werden.

Das Programm soll die eingegebene Zahl, deren Quadratwurzel und Kubikwurzel sowie die Anzahl der Iterationsschritte, die zur Berechnung benötigt wurden, ausgeben:

```
Bitte geben Sie die Zahl ein:

34

Die Quadratwurzel von 34.0 lautet 5.830951897587282


Die Kubikwurzel von 34.0 lautet 3.2396118013610797
```

Quadratwurzel

Es gibt die folgenden drei Möglichkeiten:

1. `q = x` dann ist $q^2 = n$ – der Wert der Quadratwurzel ist somit gefunden
2. `q < x` dann ist $q * x < x^2, q * x < n, x < n/q$ – für den gesuchten Wert gilt daher $q < x < n/q$
3. `q > x` entsprechend dem vorhergehenden Fall gilt $n/q < x < q$

Im zweiten und im dritten Fall wird als neuer Schätzwert das arithmetische Mittel von q und n/q verwendet. Das Verfahren wird so lange wiederholt, bis q^2 hinreichend nah an n liegt.

Kubikwurzel

Die Berechnung der Kubikwurzel erfolgt analog zur Quadratwurzel. Der neue Schätzwert wird wie folgt ermittelt: $2/3\ q + n/3q2$.

Anmerkung: Beim Einlesen der Benutzereingaben kann davon ausgegangen werden, dass nur korrekte Werte (d. h. positive ganze Zahlen) eingegeben werden.

b) Teste die Implementierung 3 Punkte

Entwickle Unit-Tests für die Implementierung.